Software defined autonomic QoS model for future Internet

Wendong Wang*, Ye Tian, Xiangyang Gong, Qinglei Qi, Yannan Hu

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

A R T I C L E   I N F O

Article history:
Received 2 March 2015
Revised 18 July 2015
Accepted 14 August 2015
Available online 20 August 2015

Keywords:
Software defined technology
QoS
Packet-marking

A B S T R A C T

Software defined technology has gained enormous momentum in both industry and academia. It may change the existing information flow architecture, which centered at hardware, by granting more privileges to deploy third-party software with more flexible network control and management capabilities. However, how to provide intelligent QoS guarantee in future network is still a hot research issue. Based on top-down design strategy, in this paper, we propose a hierarchical autonomic QoS model by adopting software-defined technologies. In this model, the network is divided into: data-plane, control-plane and management-plane. Then the data-plane is sliced as multiple virtual networks with the dynamic resource allocation capabilities, in order to carry different services or applications. The control-plane is abstracted as resource control layer and service control layer for each virtual network respectively. The management-plane then constructs hierarchical autonomic control loops in terms of different services' bearing virtual networks and the corresponding hierarchical control logic. With the hierarchical autonomic control loop and layered service management functions, this model can provide dynamic QoS guarantee to the underlying data networks. We further proposed a context-aware Collaborative Borrowing Based Packet-Marking (CBBPM) algorithm. As an important underlying QoS component in data-plane, CBBPM can be combined with other QoS components to provide some new QoS functions. Experiment comparisons between different QoS mechanisms demonstrate the effectiveness of our proposed hierarchical autonomic QoS model and the packet marking mechanism.

1. Introduction

Quality of Service (QoS) management is an important component of network management. However, being limited by the “Best Effort” principle and the connectionless transport protocol, existing IP network cannot satisfy users' different QoS demands. Nowadays, it has become an urgent issue to enhance the QoS management mechanism in future network.

Traditional Internet adopts loosely coupled architecture, which caused difficulties to realize effective control and management of the overall network resources. Based on open-loop network architecture, QoS management components apply statistical analysis to different service flows, and then formulate a corresponding QoS strategy. Diff-Serv (Blake et al., 1998) and IntServ (Braden et al., 1994) are two representative QoS models in traditional IP networks. DiffServ can be used to provide low-latency service to critical network traffics while providing simple best-effort service to non-critical traffics. Different from DiffServ, IntServ specifies a fine-grained QoS system, which contrasts to DiffServ's coarse-grained control system. IntServ allows critical traffic data to reach the receiver without interruption. However, each router in network needs to implement IntServ and each application that requires an individual reservation. In the case of a large amount of service flows, IntServ has poor extensibility.

Automatic network technology solves the original inability of combining individual QoS management components into a closed-loop to adjust operating parameters in real-time. Especially, with context awareness, QoS management components can generate context-aware QoS control loop by selectively assembling components according to network states and traffic states. And, the control loop is capable of adjusting QoS operating parameters adaptively. Compared to the traditional IP network, the autonomic based QoS control loop, to some extent, can realize higher efficiency when allocating network resources. However, sharing a common problem with the traditional IP QoS management, QoS management components in autonomic network architecture cannot be integrated flexibly. This problem constrains our ability to formulate more reasonable QoS strategies.

Software-defined technology has become the focus of both academia and industry. It also announces an era of “Software Defines Everything”. Since software is capable of integrating hardware facilities' computing and storage functions, the software-defined technology is also related to the integration of infrastructures. Software Defined Network (SDN) is an instance of Software-defined technology.
being applied in Internet (McKeown et al., 2008). In terms of storage, SDN relies on a software layer to conduct complicated management of infrastructures. While in terms of network management, SDN can realize centralized management of all network elements with a software controller without configuring each device individually. In a traditional IP network, control-plane and data-plane together provide the data forwarding and routing service. Once the forwarding path of certain traffic has been determined, the path information will be delivered to data-plane and consequently be stored in hardware. Data-plane usually selects the latest path information to forward packets. The pattern is high efficient for packet forwarding, but is fundamentally flawed in scalability. First, non-centralized management makes each network device configured repeatedly. Second, since traditional network devices are not programmable, it is difficult to realize complex management tasks (such as QoS management) through programming. With SDN controller’s centralized administration, configuration strategies are delivered in a unified way. More importantly, for certain requirement, this centralized administration is capable of developing new management components and further integrating these individual components to fulfill complicated tasks. As far as QoS management is concerned, software-defined technology makes it possible to construct a complete feedback control loop by selecting optimal QoS management components on-demand.

While SDN, as a new emerging technology, is attracting people’s attention, there are still inadequacies in QoS management within the existing SDN framework. The deficiencies are embodied in two aspects. First, few current research is dedicated to QoS management framework in SDN specification. Second, SDN framework lacks necessary supports for QoS management components. Though OpenFlow, one of the most representative instances of SDN, has been applied in many scenarios, it so far can only provide limited support to QoS management. Limited by OpenFlow switch’s architecture, OpenFlow controller can configure actions to process only specific protocols that can be recognized by the switch program, such as IP, TCP and etc. However, OpenFlow switch is unable to recognize and support any new protocol at runtime. Moreover, the set of queue configurations supported by OpenFlow switch is also quite limited. For reasons given above, researches on both QoS management framework and QoS management components in SDN are urgent.

To solve problems mentioned above, we proposed a software-defined autonomic QoS management model in this paper. By introducing autonomic network framework, the proposed model can perceive and analyze different contexts, and then generate optimal QoS management strategy. According to the optimal strategy, the model can select appropriate QoS management components to create specific feedback control loop. In this paper, we also designed a Collaborative Borrowing Based Packet-Marking (CBBPM) algorithm, which aims to improve the utilization rate of network resource. Packet marker component developed from the CBBPM algorithm can make each network device configured repeatedly. Second, since traditional IP network, ITU-T delivered a proposal Y.1291 (ITU-T, 2004) to specify the QoS management framework in packet switching network. The core concept of this framework is to combine general management components such that network services’ response can be decomposed into service requests, which may be assigned to specified network elements. QoS management components are distributed over three logic planes: control-plane, management-plane and data-plane. Fig. 1 demonstrates the QoS management framework. Each plane contains multiple QoS management components, but these components cannot be adaptively integrated as a closed control loop. Therefore, it is difficult to automatically adjust QoS strategies in real-time according to network context.

Autonomic network technologies have been gradually introduced into network management, especially the QoS management to solve the above problem. Autonomic network is a flexible architecture with dynamic and security natures. Through self-organization, it can provide various network operation strategies and QoS management mechanisms, which may be applied to part of elements or the whole network. Autonomic network based QoS management model can adapt to network environment variations and meet different users’ demands automatically.

MAPE model (IBM, 2011) is an abstraction of an autonomic system defined by IBM. Autonomic Manager and Managed Element/Resource entities are designed in this model. Entities of monitoring, analysis, planning and execution form a control loop in the Autonomic Manager. “Knowledge”, being the central of this loop, is used to drive the control loop. “Knowledge” relates to all the pieces connected by the general “knowledge” space. For example, monitoring information supplied by the sensor part of the Managed Element/Resource is part of the “knowledge”. It also includes how to analyze monitoring information, how to plan the actions to be executed as well as how and when to execute the actions on the Managed Element. “Knowledge” also includes information in the cognitive aspects of the Autonomic Manager.
Generic Autonomic Network Architecture (GANA) (EFIPANS, 2009) is another abstract model proposed by EU FP7 EFIPANS project (EFIPANS, 2009). The GANA control loop is derived from IBM MAPE model for autonomic networking management specifically. This model illustrates the possible distributed nature of the information suppliers that supply a Decision Element (DE), with information (knowledge) that is used to manage the associated Managed Element(s)/Resource(s). In this model, a Decision Element is adopted as the autonomic management element, which is similar to the role of an Autonomic Manager in the MAPE model. A Managed Entity (ME) is also adopted to represent a Managed Resource or a Managed Automated Task instead of a Managed Element. This is to avoid the confusion arising when one begins to think of an element as only a physical network element. It also illustrates the fact that behaviors or actions taken by the DE do not necessarily trigger some behaviors or the enforcement of a policy on the Managed Resource(s), but may have to do with the communication between the DE and other entities, e.g. other DEs in the system or network. For example, the DE may need to self-describe and self-advertise to other DEs in order to be discovered or to discover other DE to communicate or get “views”—knowledge known by the DE. In general, self-manageability in GANA is achieved through instrumenting network elements with autonomic DEs that collaboratively work together. GANA defines a hierarchy of DEs, i.e. four basic levels of self-management: protocol, function, node and network level. Each DE manages one or more lower-level DEs through a control loop. These lower-level DEs are therefore considered Managed Entities (MEs). Over the control loop, the DE sends commands, objectives, and policies to an ME and receives feedback in the form of monitoring information or other types of knowledge. Each DE realizes some specific control-loop(s), and therefore, represents an “Autonomic Activity” or “Autonomic Function” (AF). Examples of Autonomic Functions are Autonomic QoS Management-DE, Autonomic Security Management-DE, Autonomic Fault Management-DE, Autonomic Mobility Management-DE, etc.

Recently, several approaches have been proposed for QoS management in SDN environment. Most of these solutions are not implemented by extending existing OpenFlow protocol. (Civanlar et al, 2010) proposed a QoS routing algorithm to distinguish traffic flows with specified QoS requirements from normal traffic flows, such that flows with high priority are forwarded preferentially. As a follow-up work, (Egilmez et al., 2012) opens its APIs to users for interfering route planning procedure. Different from the two approaches stated above, (Bari et al., 2013) introduced a management layer by utilizing the north bound interface of SDN controller to validate and enforce QoS related policy, thus affecting the route planning result. These approaches all aim to provide QoS means at route selection level. However, no fine-grained QoS strategies (packet classifying, packet marking, queue managing and queue scheduling) are involved in these works.

(Kim et al., 2010) proposed an autonomic and scalable QoS architecture, which introduced rate-limiters and queue mapping APIs in OpenFlow. The rate-limiter APIs provide functionalities to map one or more flows to a specific rate limiter, while the queue mapping APIs can map a flow to a specific priority queue. According to OpenFlow version 1.0, (Ishimori et al., 2013) enhanced QoS management capability by introducing hierarchical token bucket based packet shaping as well as stochastic fairness queuing based packet scheduling. Another work we proposed before (Tong et al., 2013) designed a PindSwitch model to provide output queue controlling ability for queue configuration and queue scheduling. This paper mainly focused on how to deal with the problem of co-existence of multi-protocol in SDN switch. (Wendong et al., 2014) also proposed a QoS Management architecture. In this previous paper we took a first stab at introducing autonomic technologies into software-defined network. However, it did not go further to touch the key issues of how to develop flexible QoS management components to comply with the autonomic QoS management model. Compared to our previous works, in this paper, we further studied the hierarchical structure of autonomic QoS management model based on a top-down methodology. Also, we proposed a novel packet-mapping algorithm-Collaborative Borrowing Based Packet-Marking (CBBPM), based on which a new QoS management component could be constructed and deployed to SDN nodes. Experiments revealed the effectiveness of our proposed model and the affiliated management components.

3. Design objective

In order to enhance the flexibility of network management, especially the QoS management within a complex network environment, we design a software-defined automatic QoS management model for the future Internet in this paper. The following points are key objectives.

1. Componentized architecture in software-defined pattern

Functions provides by traditional network are inadequate in managing network service. In order to provide a dynamic, manageable, cost-effective and adaptable architecture, the network control and forwarding functions should be decoupled, such that network control is programmable and the underlying infrastructure is abstracted from applications and network services. Based on the decoupling characteristics, basic network functions and advanced functions can be composited automatically and the components management and process management can be software defined.

2. Autonomic network management

The management complexity of the Internet caused by its rapid growth becomes a major problem that limits its usability in the future. The ultimate objective of introducing automaticity is to decrease the complexity of network control and management. By introducing an autonomic control loop, a series of self-* attributes, such as self-awareness, self-configuration, self-protection, self-healing, self-optimization, and self-organization, are implemented to realize the automaticity and minimize the burden of manual operations.

3. Context-aware management enabled

Context-awareness is important for QoS policy formulation. Within this model, network context, service context and packet context are synthesized to seek optimal QoS strategy, such that network resource can be allocated effectively among multiple data flows. In this case, key packets of flows with high priority can be prioritized for delivery. Specifically, high efficient packet-marking solutions, which conform to the specifications of OpenFlow framework, would be explored.

4. Programmability

To build a new network with diverse features, it is required that the new network should be designed like software. This network architecture is programmable and it decouples the control and data forwarding. The network could also be better controlled and operated more efficiently by using software.

5. Supporting multi-heterogeneous service delivery

The network architecture should satisfy different service requirements, such as constant bit-rate real-time voice service, variable bit-rate real-time broadcast video service and variable bit-rate non real-time Internet service. According to the property of services, this architecture should provide different QoS, data transfer modes, communication protocols and different service program APIs to different services. Each service, without affecting other services, has its own service control logic and independent network architecture.
4. Architecture design

Based on the objectives we elaborate in previous section, the design principles are summarized as follows:

(1) Programmability

Networks, which are implemented according to the designed architecture, would be more efficient in network resource utilization, with open standard programmability. Network control is directly programmable because of the decoupling from forwarding functions.

(2) Autonomic

To let the network support service integration in heterogeneous network environment, we need to introduce autonomic technologies in management-plane to improve the self-managing capacity of both network and services.

(3) Virtualization

By decoupling the logic control and underlying physical infrastructures, multiple virtual networks can co-exist with no distractions. For each virtual network, a corresponding management-plane should be implemented, thus the network traffic could be flexibly controlled.

Here, we would like to further describe the detail of our proposed network architecture.

(1) Hierarchical network architecture

Considering the excellent open nature, physical network devices, which constitute the physical network, are all programmable and intelligent. It would be easy to access and control these devices and even deploy novel network protocols to fulfill specific requirements. Since control-plane and data-plane are separated, users can build their own virtual networks and define personalized network management strategies. Moreover, the decoupling of control-plane and data-plane also brings network operator the convenience to provide end-to-end controllable services to customers. In order to enforce more flexible QoS management to services with different traffic characteristics, we designed hierarchical network architecture as illustrated in Fig. 2. Generally, the physical infrastructure is virtualized as different virtual networks for the convenience of deploying different services within certain virtual network.

Based on the consideration, we designed the hierarchical architecture, which is composed of the following layers and planes:

Physical network layer: It is the data-plane that comprises physical network infrastructures, which achieve the basic connectivity of the networks. Physical network layer is the bear network of all kinds of services and applications. It also provides secure channel to support the control information communication between management plane and data plane.

Virtual networks layer: Physical resources, such as switches and servers at data-plane are first virtualized, and then these virtual resources consist of different kinds of virtual networks to support specific services and applications. These virtual networks belong to data-plane, but are virtualized in different network topology layers for different services and applications. Virtual network infrastructure is realized by gathering appropriate virtual resources from the data-plane. A new control-plane and management-plane is created to control and manage those virtual resources. These virtual networks can support different network architectures and different services, and even the relevant control and management functions can be customized according to the specified application/service carried by different virtual networks.

Network resources control layer: By using a kind of universal description of the virtual resources, each virtual network provides a universal virtual resource abstraction to the virtual network resources control layer. The control logic in the virtual network resource control layer is responsible for detecting, maintaining and allocating the virtual resources, as well as granting the virtual network resource control logic programmatic access to the virtual network. These virtual network resource control logic, such as path creation, data forwarding roles, traffic engineering and etc. determine the designed virtual network behaviors. Note that, since each virtual network typically carries different services, the control logic of different virtual network resources control layer can be different.

In order to support the virtual network resource controlling, this layer is further divided into two sub-layers: basic switch function sub layer and basic network function sub layer. Generally, basic switch function sub layer is composed of protocol description management module, queue resource management module, action management module and flow-table management module. The core functions that basic switch function sub layer performs are mainly reflected in aspects of: (1) Data link layer protocol analyzing, based on which switch is able to transmit and receive data encapsulated with different data link layer specifications. (2) Protocol parsing, based on which switch conducts field extracting and packet analyzing with any protocol format. (3) Flowtable and actions management. (4) Queue resource management, through which the queue related operations, such as queue management and queue scheduling, are implemented. Virtual basic network function sub layer is responsible for global network management. Generally, this layer is composed of network monitoring and management module, topology discovery and management module, routing management module as well as resource management module. Functions performed by this layer are independent with the underlying physical infrastructures. Actually, basic switch function sub layer can be regarded as a managed entity of its upper layer according to the perspective of GANA reference model.

Service control layer: When the specific services or applications are deployed on a virtual network, the dedicated service control logic would be constructed automatically to control and manage the service deployment according to its specific requirements.

Management-plane: There are two types of management-plane: Generic Network Management (GNM) plane and Virtual Network Management (VNM) sub-plane. GNM is a generic autonomic manager (or called Decision Element (DE)), which manages all VNMs. While each VNM could be considered as a Managed Entity (ME) in GNM plane, those VNM MEs together with GNM DE consist of an autonomic system on GNM plane.

GNM is responsible for managing all VNMs. It also performs the function of physical resource configuration and physical network organization. Working with all VNMs, GNM could implement several self-* attributes, such as self-configuration, self-organization and self-optimization.

VNM is responsible for managing virtual networks, virtual resources and associated services or applications. VNM is defined as an autonomic manager (or DE) for virtual networks. The virtual networks layer, virtual network resources control layer and service layer could be considered as its MEs.
(2) Key solutions
To realize the proposed architecture, the following mechanisms are indispensable. First, context awareness should be introduced to dynamically allocate virtual resources among different virtual networks. On this basis, virtual networks are partitioned through rationally combining virtual resources. Actually, virtual networks are isolated from each other, in order to realize effective management; autonomic control loop is deployed in each virtual network. The following sections give descriptions of the key mechanisms in detail.

A. Context-aware based virtual resource redistribution
Since multiple virtual networks share same network resources (such as network bandwidth, CPU, queue, and etc.), it is necessary to make a sound resource allocation mechanism to ensure the overall equity and effectiveness. Traditionally, each virtual network in SDN is allocated with a static and fixed share of network resource. However, this approach is of low efficiency in actual deployment. For example, service may be experienced with high end-to-end delay in a virtual network, while other virtual networks are still lightly loaded. The direct reason is that the bandwidth of each substrate link is ensured by resource isolation. Therefore, to address this problem, generic network management plane functions to dynamically allocate network resources between various virtual networks via context-based reasoning (The detail procedure is elaborated in Section VI). In general, the overall resources of underlying physical network infrastructures are sliced according to the requirements of each service or application.

As illustrated in Fig. 3, an autonomic control loop, this is comprised of four connected components: monitor entity, analyze entity, plan entity and execute entity, is implemented on the GNM plane. The monitor entity perceives the environment of physical and virtual networks (e.g. the state of the networks and available resources) as well as the resources demanded from the services running on virtual networks in real-time. Then, the analyze entity applies comprehensive analysis according to the remained network resources, service requirements in different virtual networks and the network’s overall objectives. With appropriate inference model, plan entity generates policies to specify how to adjust the network resources. Finally, the execute entity carries on a series of operations to complete the resource redistribution among multiple virtual networks. Within the framework of autonomic control loop in GNM, virtual resources management is achieved by the network itself through the autonomic attributes, such as self-configuration, self-organization and self-optimization of the inter-virtual-network resources, which maximize the utility of the network resources.

B. Network virtualization
Network virtualization is the key approach to implement the sliced virtual network on a common underlying physical network. Basically, two key mechanisms are involved in this task: network resource isolation and virtual network management. For the former, we proposed an extensible packet matching and pipeline processing based mechanism, which is applied on the data-plane. For the latter, we designed a virtual network management component (VNMC), which is located on the management-plane.

According to the specification of OpenFlow version 1.3, the extensible packet matching and the pipeline feature could be used to provide more flexible network virtualization mechanism. In order to support multiple forwarding paradigms, all flow-tables should be designed to support the extensible packet matching mechanism. The OpenFlow pipeline is also divided into multiple sub-pipelines. The number of sub-pipelines is equivalent to the number of virtual networks that were designed for multiple forwarding paradigms.

Pipeline processing always starts at the first flow-table, and every packet is first matched with entries of flow table 0. Depending on the matching results in the first table, other flow-tables would then be used. Thus, in our approach, the flow-table 0 is used as a de-multiplexer, which dispatches packets and flows to different sub-pipelines. Fig. 4 illustrates a specific case of the proposed mechanism, where the switch carries flows of three virtual networks. As shown in Fig. 4, different kinds of forwarding approaches could be used in different virtual networks. For example, the traditional IP-based packets routing and forwarding mechanism is used in virtual network 1, while the virtual network 2 and 3 are used to support label-based switch/routing and named-based routing/forwarding respectively.

Regarding to the management-plane, we designed a virtual network management component (VNMC) to apply network resource allocation and other virtual network management relevant tasks.

As illustrated in Fig. 5, VNMC works above the virtual network operations system (NOX1.1 Core Platform), the main objectives include:
(1) establishment, modification and destruction of virtual network. 
(2) Network resource perception and network resource allocation. 
The two tasks correspond to the key functions of management-plane 
described in previous section.

Above the virtual network management component (VNMC) is 
various virtual slices. Each virtual slice corresponds to a specified virtual 
network. Virtual slice is composed of multiple control modules, 
which function as decision elements of the virtual network and the 
services that are running on the virtual network.

C. Autonomic network structure

In order to improve the capacity of supporting adaptive management 
on both customer service and network infrastructures, a series of automatic control loops are introduced into the management-plane. 
Actually, the linkage mechanism by integrating the four entities enables the model to automatically adjust the configurations of network parameters by analyzing the collected contexts from different layers. Monitor entity functions as a context collector to perceive and organize context from required managed resources. By analyzing the collected contexts, analyze entity produces sets of control policies. Then the plan entity would be activated to decompose the scheduled control policies into a series of operations. Finally, the execute entity is responsible of putting the operations into practice via configuring settings on physical network elements (Fig. 6).

Three control loops indwell in a certain virtual network management-plane. Each of the three control loops is associated with one of the three layers: virtual network layer, network resource control layer and service control layer. For virtual network layer, the control loop is in charges of the autonomic attributes of virtual network nodes. In particular, the monitor entity in the control loop monitors the information such as the size of flow-table, the utilization of CPU and the queue length. And the monitor entity on network resource level monitors the data such as virtual network topology, traffic load, available bandwidth and etc. While the monitor entity on service control layer aims to grasp the context about service-relevant information, such as service type, service parameters, service QoS requirements, service performance and etc. By aggregating all the raw context data, analyze entities indwelling in each layer build models to capture the changes of network environment and determine whether any actions should be carried on. Based on the policies generated by analyze entity, plan entities then make decisions to guide various self-reconfigurations to dynamically adjust the network environments. Here, the policies would be regarded as a set of rules, and each of which is composed of a set of conditions and actions.

Once a certain condition is satisfied, the corresponding set of actions would be triggered. Policy based network management, a systematic network management system, emphasizes that the self-* attributes should be inherent and the virtual resources utilization within a virtual network should be able to optimize by themselves. Note that, the three control loops are not isolated but interconnected, such that the contexts collected in certain layer could be shared. Moreover, the policies and actions that are generated in different layers may be interdependent.

5. QoS management components

To guarantee the quality of services, QoS management components are enabled on both the management-plane and the data-plane. Specifically, QoS management DE, which determines QoS rules dynamically, is located on the management-plane as an application. Meanwhile, the basic capability of QoS components in data-plane is also extended.

Generally, QoS management DE is composed of six modules. The functions of each module are elaborated in Fig. 7.

- **Requirement DB**: a database, which stores the detailed QoS requirements of each data flow. Actually, this information is translated from Service Level Agreement (SLA), which has been made between the user and network service provider.
- **Rule DB**: a database in which QoS rules specially planned for certain network environment or specific QoS requirements are stored.
- **Policy DB**: a database in which global QoS policies (such as the working mode of packet marker) are stored.
- **Context manager**: an entity, which is responsible for analyzing the context information perceived from the data-plane. Generally, here the context refers to any information that characterizes the status of network, service and packet. Specifically, network context is defined as a set of information, which characterizes the performance of network nodes (e.g., utilization of CPU, memory of node and the queue length) and link status (e.g., the packet loss ratio, delay, jitter and the available bandwidth). Service context is defined as a set of information, which depicts the inherent features (e.g., the service type and the QoS requirements) and real-time features (e.g., the burst rate) of a service flow. The inherent features are carried in the first packet of flow. They could be extracted by leveraging Deep Packet Inspection (DPI) technology, whereas the real-time features are obtained through statistical analyze. The packet context refers to the semantic priority of each packet within a flow, and packets with different priority have different impacts on a certain service in terms of quality of experience.
- **Analysis**: this module is responsible for judging whether the QoS requirements of a service could be satisfied and whether conflicts exist among these requirements. Once the QoS requirements are satisfied, they along with the corresponding contexts are delivered to the QoS rule decision module.
- **Rule decision**: if appropriate QoS rules exist, this module would select an optimal rule for service. Otherwise, rule decision module
Collaborative borrowing-based packet-marking are added into the content layer of NetConf. This protocol needs to be enriched to provide more operation sets via management and queue scheduling, the content layer of NetConf are far from fulfilling QoS functionalities. Therefore, in order to support abundant means to diverse packet marking options, it is necessary to extend the existing meter band. Since fields in meter band are not fixed, in order to avoid the inconvenience caused by unawareness of the structure of a new meter band in advance, type-length-value format (as shown in Fig. 8) is explored to define the newly introduced meter band.

The queue of SDN switch is configured via OF-Config, which is transported with NetConf protocol. Current available operations, such as <get-config> and <edit-config> could only provide basic functionalities to read or modify (set the minimum or maximum transmission rate of a queue) the configurations. In order to support queue management and queue scheduling, the content layer of NetConf protocol needs to be enriched to provide more operation sets via OF-Config. Thus in our model, Weighted Random Early Detection (WRED) queue management algorithm and Priority Queuing (PQ) as well as Weighted Round-Robin (WRR) queue scheduling algorithms are added into the content layer of NetConf.

### 6. Collaborative borrowing-based packet-marking

Packet-marking is a key mechanism in QoS management. Once network congestion occurs, network nodes would selectively discard packets according to its flag. In this section, we would elaborate the proposed Collaborative Borrowing Based Packet-Marking (CBBPM) algorithm that complies with the concept of software-defined autonomous QoS model. Being compatible with the traditional packet marking algorithms, our proposed approach possesses a variety of autonomous abilities as stated below:

1. CBBPM can automatically perceive network context, service context and packet context.
2. CBBPM can automatically configure corresponding policy and parameters of packet marker according to the perceived service type and performance data.
3. CBBPM can provide semantic-level differentiated services to different data flows, and bring differentiated marking to packets with different semantic priority.

### A. Design principle

Traditional Single Rate Three Color Marker (srTCM) is used as a key component in DiffServ traffic conditioner (Heinannen & Guerin, 1999). srTCM meters a traffic flow and marks its packets to be either green, yellow or red according to three traffic parameters: Committed Information Rate (CIR), Committed Burst Size (CBS) and Excess Burst Size (EBS):

- **Committed Information Rate (CIR):** an index which defines the permitted average rate of a flow. It also represents the rate of token injection.
- **Committed Burst Size (CBS):** a maximum volume of token which is allowed for every burst of each flow. It also represents the maximum rate of destruction of token.
- **Excess Burst Size (EBS):** a maximum volume of token which is allowed for exceeding CBS for every burst of each flow.

A packet is marked as green if it does not exceed the CBS, yellow if it does exceed the CBS but not the EBS, and red otherwise. The packet marking behavior depends on both operation modes (i.e., Color-Blind and Color-Aware, Color-Blind mode is adapted in this paper) and two token buckets: C and E. The two token buckets share the common rate CIR. The maximum size of C token bucket and E token bucket are CBS and EBS respectively. Initially, the two token buckets are full. Then, the number of tokens is updated per 1/CIR second according to the following rules:

- If \( T_c \) (the number of tokens left in C token bucket) is less than CBS, then a token is injected into C token bucket.
- If \( T_e \) (the number of tokens left in E token bucket) is less than EBS, then a token is injected into E token bucket.
- Neither token account in C token bucket nor E token bucket is increased.

In color-blind mode, packets are assumed to be uncolored. When a packet \( P \) arrives at time \( t \), it would be handled as follows:

- If \( T_c \) is greater than P.size (the size of packet \( P \)), packet \( P \) is marked as green and \( T_c \) is decremented by P.size,
- Else if \( T_e \) is greater than P.size, packet \( P \) is marked as yellow and \( T_e \) is decremented by P.size,
- Else packet \( P \) is marked as red, neither \( T_c \) nor \( T_e \) is decreased.

The srTCM process flows independently. However it is liable for the waste of tokens considering the situation that some flows may discard surplus tokens if they are under light load, while other flows with heavy load are utterly lacking of enough tokens. Furthermore, even within a flow, packets with different priority may be of different importance to service. However, traditional packet marking algorithms do not process these packets with the consideration of their different semantic priorities. To achieve better resource management and allocation, it is necessary to improve the existing packet marking algorithms by introducing autonomic technology, such that the quality of service of key packets would be enhanced.

![Fig. 8. Format of meter band with variable specific arguments.](image)

<table>
<thead>
<tr>
<th>Band Type</th>
<th>Rate</th>
<th>Counters</th>
<th>Spec_arg #1</th>
<th>Spec_arg #1</th>
<th>Spec_arg #1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Type</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spec_arg #n</td>
<td>Spec_arg #n</td>
<td>Spec_arg #n</td>
<td>Spec_arg #n</td>
<td>Value</td>
<td></td>
</tr>
</tbody>
</table>

![Fig. 9. illustrates the basic concepts of CBBPM algorithm.](image)
In CBBPM algorithm, packet marker applies marking operation to packets according to two categories of parameters: service related parameters and policy related parameters. Service related parameters are those statistical indexes, which depict traffic characteristics. Herein, they refer to CIR, CBS and EBS. While policy related parameters determine the global policy of packet marker. Generally, the following policy related parameters are used:

- **Packet dropping profile**: it specifies the maximum number of packets, which are tolerant to be dropped continuously. It is denoted as \( L \).
- **Ratio of key packet**: it specifies the ratio of key packets in all packets, denoted as \( r_{cri} \).

### B. Definition of context

Three types of context are involved in this packet-marking algorithm. Detailed descriptions of the contexts are given below:

- **Service context**: it refers to the set of information, which reflects the attributes and status of current flow. Service context is collected by context agents, which reside in autonomic terminals. Service context is encapsulated within the QoS option of packet header. Generally, the following types of service context are considered in CBBPM algorithm.

#### i. Borrowing context

- **Ratio of key packet**
- **Network context**: it refers to the set of information, which describes the status of current network. Out of the many parameters depicting network status, the average PLR\(_{path}\) (packet loss rate per path) of end-to-end link reflects the congestion level statistically. Thus, the value of PLR\(_{path}\) could be regarded as the indicator for adjusting the token allocation policy. Once PLR\(_{path}\) arises, the congestion level increases. Thus the loss rate of non-critical packets needs to be increased appropriately to guarantee the success of transmitting critical packets. Actually, the value of PLR\(_{path}\) approximately equals to the estimates of PLR\(_{flow}\), and the actual value of PLR\(_{flow}\) could be derived from QoS option field of IPv6 packet header. To better reflect the actual packet loss rate on network link, Exponentially Weighted Moving Average (EWMA) means is adopted to calculate PLR\(_{flow}\) according to the historical PLR\(_{path}\) and current PLR\(_{flow}\):

\[
PLR_{path} = \alpha \cdot PLR_{path}^t + (1 - \alpha) \cdot PLR_{flow} \quad 0 \leq \alpha \leq 1
\]

where, PLR\(_{path}\) denotes the historical value, while PLR\(_{flow}\) is the feedback value extracted from IPv6 packet header. The smooth factor \( \alpha \) balances the weight of PLR\(_{path}\) and PLR\(_{flow}\).

- **Packet context**: it defines the semantic priority of a certain packet within its subordinate flow. Context agent, which is running on sending terminal, firstly recognizes the semantic priority of a packet and then marks it with corresponding precedence level in Traffic Class (TC) field. For data flows with \( K \) different precedence levels, we define a packet as critical packet if its precedence level \( k \) satisfies the condition that \( 1 \leq k \leq K_0 \), where \( K_0 \) is a predefined parameter. Packets with precedence level \( k \) \( (K_0 \leq k \leq K) \) are defined as non-critical packets.

### C. Algorithm description

In CBBPM algorithm, three token buckets (\( C_{cri} \), \( C \) and \( E \) token bucket) are involved for each data flow, where the specification of C token bucket and E token bucket follows the definition in srTCM. As a newly introduced token bucket, tokens contained in \( C_{cri} \) are just allocated to non-critical packets. We define the bucket size of \( C_{cri} \) as \( r_{cri} \), CBS and the rate of token injection is \( r_{cri} \) CIR. \( r_{cri} \) is adopted to represent the number of tokens remained in \( C_{cri} \) token bucket. To better characterize the status of packet marker, \( n_g \), \( n_y \), and \( n_r \) are defined to record the total bytes of packets, which are marked as green, yellow and red respectively. The purpose of introducing the variables of \( n_g \), \( n_y \), and \( n_r \) is to guarantee the fairness and to avoid the situation of excessive amount of packets being marked as red.

CBBPM algorithm can be decomposed into three phases: token updating, packet marking and adaptive adjustment. Generally, during token updating phase, tokens are injected to three token buckets periodically. During packet marking stage, packets are marked as green, yellow or red according to perceived contexts and configured policy. Adaptive adjustment phase aims to protect critical packets by selectively degrade non-critical packets if congestion occurs. We would give a detailed description to each stage in the following:

#### i. Token updating

In initial state, the three token buckets (\( C \), \( E \) and \( C_{cri} \)) are all full of tokens. Hereafter, the system would inject tokens to the three buckets periodically. If committed information rate were set to CIR byte/s, the system would inject a token to C token bucket and E token bucket per 1/CIR second. Meanwhile, the system would also inject a token to \( C_{cri} \) token bucket per \( 1/(r_{cri} \cdot CIR) \) second. The procedure of token update is described in Fig. 10.

#### ii. Packet marking

To provide differentiated service, critical packets and non-critical packets are processed discriminately in this stage. In order to
improve the possibility that tokens could be allocated to critical packets, a collaborative token allocating concept is introduced to improve the existing stTCM algorithm. According to the collaborative token allocating concept, if the tokens remained are not enough for a critical packet, system would regulate tokens from other flows to meet its requirement. However, for non-critical packets, the collaborative mechanism takes effect only if the total length of packets which are marked as non-conformed exceeds $L^B$, where $B$ denotes the average length of packets within a data flow. The significance of collaborative token allocation lies in: (1) by scheduling the tokens globally among flows, the possibility of marking critical packet as conformed (green) is bounded. (2) The total length of packets that are marked as non-conformed (red) should be less than $L^B$.

Suppose that $S$ flows exist and each flow maintains its own packet marker respectively. In order to record the loan relationship of tokens between different flows, two matrices, i.e., $CMatrix$ and $EMatrix$ are defined. $CMatrix[i][j]$ records the number of tokens that flow$_i$ borrows from flow$_j$’s C token bucket. Similarly, $EMatrix[i][j]$ records the number of tokens that flow$_i$ borrows from flow$_j$’s E token bucket. Once a packet arrives at a certain edge node, the packet marker would first extract its precedence level from the Traffic Class field of IPv6 header and then judge whether it is a critical or non-critical packet. The marking procedure for critical packets is described in Fig. 11.

The function $CRegain(i, P)$ is designed for recycling tokens that were borrowed from flow$_i$’s C token bucket. To avoid the potential spillage, the maximum amount of tokens that flow$_i$ expects to regain is limited to $P_{size} - T_c$. The algorithm of $CRegain(i, P)$ is described in Fig. 12.

In the procedure described above, flow$_i$ firstly determines if the amount of tokens on loan is enough to fill the gap of insufficient tokens. If the amount is less than flow$_i$’s expectation, returning the loans is the priority for the flows, which still owe flow$_i$. In this case, if the amount of tokens remained in debit side’s C token bucket is not enough to repay the loan in full, only the remained tokens are repaid to flow$_i$. Otherwise, the amount of tokens each debit should return is determined according to the ratio of its loan to the total amount of tokens that flow$_i$ has ever loaned out. If the number of remained tokens is less than the amount it should undertake, only the remained tokens are repaid back.

The function $CBorrow(i, P)$ is designed to realize the functionality of borrowing tokens from other flow’s C token bucket for the purpose of marking critical packet as green. Two conditions must be satisfied for a flow to be selected as creditor. First, adequate tokens are available in this flow’s C token bucket. Second, payload of this flow is relatively light, such that there would not be

---

**Fig. 10.** The procedure of token updating.

**Fig. 11.** The procedure of critical packet marking.

**Fig. 12.** The procedure of regaining token from critical packet’s C token bucket.
### Adaptive adjustment

Since queue management mechanisms (such as RED, WRED) would determine the dropped packet according to the congestion level of network and the packet’s marked result, to provide better quality of service to critical packets, it is feasible to dynamically adjust the relative proportion of critical packets and non-critical packets to reduce the possibility of dropping the critical packets. By degrading the non-critical packets, the proportion of critical packets among the conforming packets (marked in green or yellow color) would be relatively enhanced. Once PLRpath arises, the congestion level of network link becomes sharper, so the preventative measures of degrading non-critical packets would consequently improve the possibility of successful delivering critical packets.

---

#### Procedure: C Borrow(i,P)

1. Flow Stream = loadRankγ;
2. num=T[i];
3. for (j=1;j <=num;j++)
4. if (T[j]<>CBS(i))
5. index = map(stream);
6. if (T[index]>δ)
7. \[T[index] = T[index] - \left(\delta - load \right)\];
8. T[i]=T[i] + \left(\delta - load \right);\]
9. CMatrix[index][i]=CMatrix[index][i]+\left[\delta - load \right];
10. }
11. }
12. else Break;
13. }
14. return T[i]-num;
15. }

---

#### Procedure: Marking_NormalPkt (Flow i, Packet P)

1. if (T[i]<>P.size) \{ 
2. if ((n+1)/B>辽×(1/PLRpath-1)) \{ 
3. packet P is marked as RED;
4. n:=n + P.size, n_r:=0, n_p:=0;
5. }
6. else \{ 
7. packet P is marked as GREEN;
8. T[i]=T[i] - P.size;
9. n:=n + P.size, n_r:=0;
10. } \}
11. else if (T[i]>P.size) \{ 
12. if ($(n+1)/B>辽×(1/PLRpath-1)) \{ 
13. packet P is marked as RED;
14. n:=n + P.size, n_r:=0, n_p:=0;
15. }
16. else \{ 
17. packet P is marked as YELLOW;
19. n:=n + P.size, n_r:=0;
20. }
21. else \{ 
22. packet P is marked as RED;
23. n_r:=0, n:=0,n_p:=n_p + P.size; \}
24. }

---

#### Procedure: Degrade (Packet P)

1. if (ColorOf(P) == Green) \{ 
2. T[i]=T[i] - P.size;
3. packet P is marked as RED;
4. n:=n + P.size, n_r:=0, n:=n_p + P.size;
5. }
6. else \{ 
7. if (ColorOf(P) == Yellow) \{ 
8. T[i]=T[i] - P.size;
9. packet P is marked as RED;
10. n:=n + n_p + P.size; \}
11. }

---

### Fig. 13. The procedure of borrowing token from critical packet’s C token bucket.

### Fig. 14. The procedure of packet marking of non-critical packet.

### Fig. 15. The procedure of degrading of non-critical packets.

---

In particular, those non-critical packets marked as green or yellow would be degraded as red. Various schemes could be adopted to apply this adjustment. PLRpath is utilized as a key factor in our approach. Concretely, if the condition PLRpath < (1 – \(r_g\)) is satisfied, the packet need not to be degraded, and if PLRpath > (1 – \(r_{crit}\)), the degrade operation would be performed under probability \(P_1\), otherwise the degrade operation is applied under probability \(P_2\). Here, \(r_g\) denotes the proportion of packets marked as green to the total packets in current time slice, and \(r_{crit}\) denotes the average rate of critical packet to CIR. The detailed description of packet degrading is illustrated in Fig. 15.
7. Prototype and experiment

A test bed is constructed to verify the autonomic QoS management performance of our proposed model. Controller is implemented based on NOX platform (Gude et al., 2008) with extending Proto-table Management DE, QoS Management DE and other Des, which are introduced in section IV. While the switch is implemented based on Ofsoftswitch13 (CPqD, 2012), and the pipeline processing functionality is deployed in compliance with OpenFlow Switch 1.3. Meanwhile, actions and match fields are extended in order to support generic processing functions and user-defined OXM_TLV.

A. Testbed establishment

The test bed is established as demonstrated in Fig. 16. More specifically, three Dell R410 servers and four R710 servers are deployed in this network. One Dell R710 server running NOX platform is designated as controller, while other servers running Ofsoftswitch13 act as either core switches or edge switches. All switch nodes are DiffServ enabled. Besides, several other PCs serving as autonomic terminals are connected to the edge switches.

Three videos with different characteristics provided by the Faculty of Electrical Engineering and Computer Science, Technische Universität Berlin are selected as testing data. All of the videos are coded and encoded in H.264/AVC JM 18.5. Meanwhile, the three videos are all in Common Intermediate Format (CIF, $352 \times 288$ solution, 30fps). For each video, the semantic precedence of I frame, P frame and B frame satisfies the condition $PI > PP > PB$. In addition, the frame order of H.264 is IBPBPB…, and the interval between two I frames is 20 frames. Table 1 lists the statistical features of the three videos.

B. Verification of QoS supporting components

Three virtual networks are first abstracted from the underlying network infrastructures. Different QoS components are assembled in the three virtual networks to verify the software-defined characteristics of our proposed model. The first virtual network is derived from the switch ES1, CS1, CS2 and ES3. The second virtual network is derived from ES1, CS1, CS2 and ES4, while the third virtual network is derived from ES2, CS1, CS2 and ES4. Three virtual networks adopt the same flow classification policy, i.e., video flows are all mapped to AF class and the background traffic is mapped to BE class. Meanwhile, PQ+WRR combination is enabled in all three virtual networks as queue scheduling mechanism, while different packet-marking algorithms are configured as meter band in three different virtual networks. In virtual network 1, srTCM is enabled. CAPM (Xiangyang, 2011) and CBBPM are configured as meter band in virtual network 2 and virtual network 3 respectively. Table 2 lists the traffic related features of the three videos. WRED related parameters are listed in Table 3. Bandwidth of the bottleneck between two core nodes, i.e., CS1 and CS2 is set as 20 Mbit/s, and the delay is set as 5 ms. The bandwidth and delay of any other links are set as 100 Mbit/s and 2–10 ms, respectively. By injecting background traffic with different flow rate, we could acquire the contrast effect when applying different packet marking algorithms in three virtual networks.

Fig. 17 illustrates the average PSNR (Peak Signal to Noise Ratio) of each video, which is delivered in three virtual networks with srTCM, CAPM and CBBPM packet marking algorithm enabled respectively. Obviously, the statistical results demonstrate CBBPM’s superior performance to CAPM algorithm and srTCM algorithm. Fig. 18 shows the distribution of packet-marking results of three video streams with different packet-marking algorithms under different background traffic. As could be seen, for a given video, the proportion of green-marked packets, which delivers I frame by CBBPM algorithm, far surpasses that of the other two packet-marking

---

**Table 1**

<table>
<thead>
<tr>
<th>No.</th>
<th>video</th>
<th>frames</th>
<th>AR</th>
<th>ARcri</th>
<th>Burstiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bridge_far</td>
<td>300</td>
<td>139</td>
<td>86</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Waterfall</td>
<td>300</td>
<td>266</td>
<td>151</td>
<td>Low</td>
</tr>
<tr>
<td>3</td>
<td>Tempete</td>
<td>2000</td>
<td>254</td>
<td>65</td>
<td>Middle</td>
</tr>
</tbody>
</table>

a: AR – the average information rate of the stream (KB/s).
b: ARcri – the average information rate of critical packets (KB/s).

**Table 2**

<table>
<thead>
<tr>
<th>Video</th>
<th>CIR (Kbit/s)</th>
<th>CBS (bytes)</th>
<th>EBS (bytes)</th>
<th>frr</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bridge_far</td>
<td>136</td>
<td>3750</td>
<td>5500</td>
<td>0.57</td>
</tr>
<tr>
<td>Waterfall</td>
<td>456</td>
<td>14,000</td>
<td>28,000</td>
<td>0.64</td>
</tr>
<tr>
<td>Tempete</td>
<td>126</td>
<td>20,500</td>
<td>40,100</td>
<td>0.26</td>
</tr>
</tbody>
</table>

**Table 3**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Green packet</th>
<th>Yellow packets</th>
<th>Red packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum packet length ($th_{min}$)</td>
<td>60</td>
<td>35</td>
<td>20</td>
</tr>
<tr>
<td>Maximum packet length ($th_{max}$)</td>
<td>90</td>
<td>70</td>
<td>45</td>
</tr>
<tr>
<td>Maximum possibility of packet loss rate ($p_{max}$)</td>
<td>0.01</td>
<td>0.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Fig. 17. The average PSNR of each video.
(i) The rate of background traffic flow is 10Mb/s  
(ii) The rate of background traffic flow is 20Mb/s

---

**Fig. 18.** The distribution of packet-marking results.

---

**Fig. 19.** PSNR of 300 continuous frames under different background traffic.

---

algorithms. In general, the cause is clear: CBBPM algorithm could dynamically adjust the resource allocation according to real-time context, such that the resource utilization is maximized. To take a deeper look at the change of PSNR of each video over time, 300 continuous frames are selected. The results are demonstrated in **Fig. 19.**

As demonstrated in **Fig. 19,** the PSNR of three videos, which are processed with CBBPM packet marking algorithm, exceeds the results that achieved under the other two algorithms in different jamming environments. Furthermore, certain frames of these videos are picked out as **Fig. 20.**

---

Obviously, from the procedures stated above we can conclude that within our proposed software-defined autonomic QoS model: (1) Various components could be assembled dynamically in software-defined way to meet different QoS management requirement in different virtual networks. (2) Autonomic QoS management could be realized by obeying the autonomic network structure. (3) By adopting appropriate QoS mechanisms in software-defined environment, network resource (i.e., token) could be allocated among different parties in different levels of granularity.
8. Conclusion

In this paper, a novel software-defined automatic QoS management model for future network has been proposed. Based on top-down design strategy, autonomic technologies and GANA architecture reference model have been introduced for remodeling data-plane, control-plane and management-plane. This model provides a universal scheme for autonomic network management in SDN. Furthermore, a context-aware Collaborative Borrowing Based Packet-Marking (CBBPM) algorithm is proposed. As an important underlying QoS component in data-plane, it can be combined with other QoS components to provide some new QoS functions. Experiments demonstrated the effectiveness of this model. In the future, we hope that further investigations and deep analysis could be continued, and a demo development is also necessary to prove our design.

Acknowledgments

This work was supported in part by the National High Technology Research and Development Program (863 Program) of China under grant no. 2011AA01A101, the EU FP7 Project EFIPANS (INFOSO-ICT-215549), the National Natural Science Foundation of China under grant nos. 61370197 and 61271041.

Reference


EFIPANS, 2009. Exposing the Features in IPv6 protocols that can be exploited/extended for the purposes of designing and build autonomic Networks and Services project, at http://efipsans.org (accessed 15.10.10.).


Wendong Wang is a full professor of State Key Lab of Networking and Switching Technology at Beijing University of Posts and Telecommunications. His research interests are IP QoS, next generation Internet, and next generation Internet services.

Ye Tian received his PhD degree from Beijing University of Posts and Telecommunications in 2013, he is now a Post-doctoral in State Key Lab of Networking and Switching Technology. His research interests are IP QoS, next generation Internet services and data mining.

Xiangyang Gong is a full professor of the State Key Lab of Networking and Switching Technology at BUPT. His research interests are IP QoS, network security, advanced switching technologies and next generation Internet. Xiangyang Gong received his PhD communication science from Beijing University of Posts and Telecommunications.

Qinglei QI is currently a PhD candidate student of the State Key Lab of Networking and Switching Technology at BUPT. His research interests are IP QoS, Software-defined Network.

Yannan Hu is currently a PhD candidate student of the State Key Lab of Networking and Switching Technology at BUPT. His research interests include next generation Internet, Software-Defined Networking and network optimization.